



**Leibniz Supercomputing Centre**  
of the Bavarian Academy of Sciences and Humanities

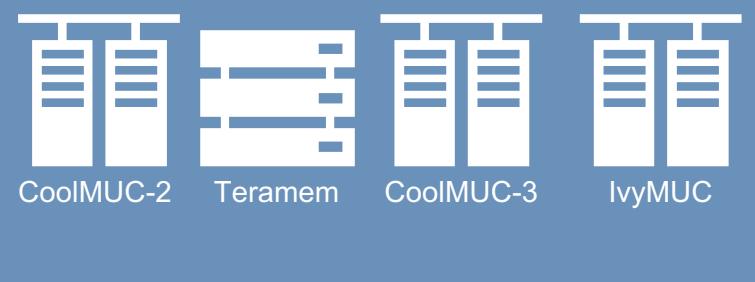
# Deep Learning on the LRZ AI Systems

# An High Level Overview of Some LRZ Resources



DSS

(Data Science Storage)



lxlogin8.lrz.de

lxlogin[1-4].lrz.de

lxlogin10.lrz.de



LRZ AI Systems



Compute Cloud

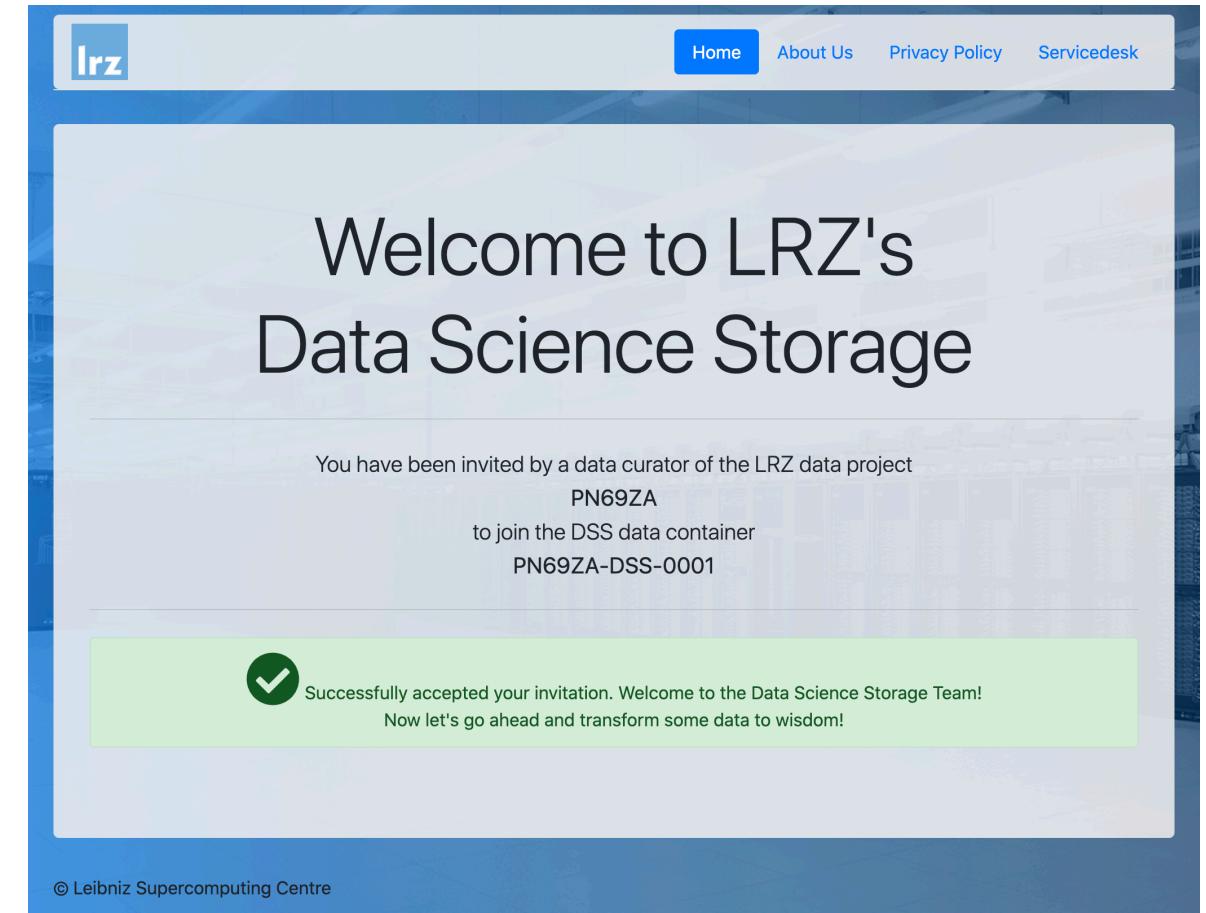


AI Ready

# The LRZ AI System

# Who can access?

- User requirements to get the access:
  1. Own a Linux Cluster account,
  2. Send a request through a service request ticket explaining the intended use.
- Upon approval, you will be invited to a DSS container - You need to accept this invitation before being able to access the AI systems!
- This DSS container will be used as your \$HOME (although this is going to change in the future.)



# Resources Overview

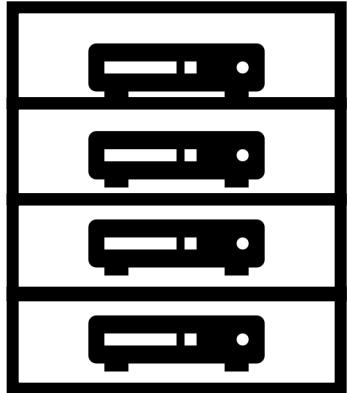
	<b>Slurm Partition</b>	<b>Number of nodes</b>	<b>CPUs per node</b>	<b>Memory per node</b>	<b>GPUs per node</b>	<b>Memory per GPU</b>
<b>DGX A100 Architecture</b>	lrz-dgx-a100-80x8	4	256	2 TB	8 NVIDIA A100	80 GB
	lrz-dgx-a100-40x8	1	256	1 TB	8 NVIDIA A100	40 GB
<b>DGX-1 V100 Architecture</b>	lrz-dgx-1-v100x8	1	80	512 GB	8 NVIDIA Tesla V100	16 GB
<b>DGX-1 P100 Architecture</b>	lrz-dgx-1-p100x8	1	80	512 GB	8 NVIDIA Tesla P100	16 GB
<b>HPE Intel Skylake + NVIDIA Node</b>	lrz-hpe-p100x4	1	64	256 GB	4 NVIDIA Tesla P100	16 GB
<b>V100 GPU Nodes</b>	lrz-v100x2 (default)	4	20	368 GB	2 NVIDIA Tesla V100	16 GB

# Using the Cluster

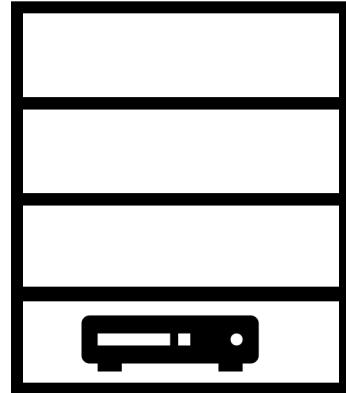
- SLURM: Simple Linux Utility for Resource Management
  - open source
  - fault-tolerant
  - highly scalable
- Cluster management and job scheduling
- (Three) main tasks
  - allocates exclusive and/or non-exclusive access to resources (compute nodes)
  - provides a framework for starting, executing, and monitoring work on the allocated nodes
  - arbitrates contention for resources by managing a queue of pending work.



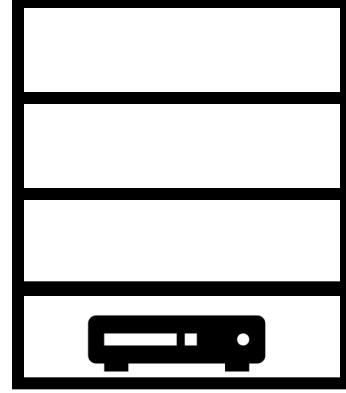
# LRZ AI System Configuration



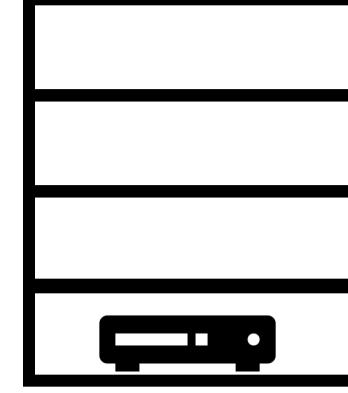
lrz-dgx-a100-80x8



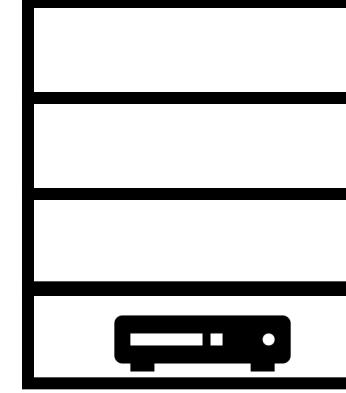
lrz-dgx-a100-40x8



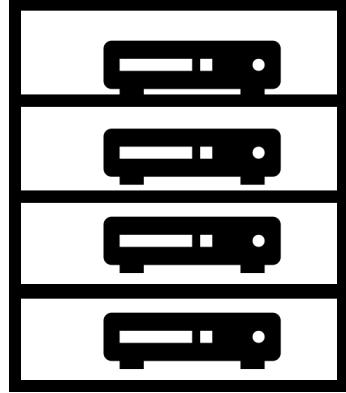
lrz-dgx-1-v100x8



lrz-dgx-1-v100x8



lrz-hpe-p100x4



lrz-v100x2



Private  
Network



datalab2.srv.lrz.de



datalab3.srv.lrz.de



MWNet

# Accessing the LRZ System

- Login node datalab2.srv.lrz.de accessible via ssh

```
ssh -Y datalab2.srv.lrz.de -l xxyyzz
```

- From the login node, jobs are submitted to the hardware described at the beginning of this course using SLURM
- A couple of handy SLURM commands

```
$ squeue
```

```
$ sinfo
```

```
$ salloc /  
scancel
```

```
$ srun
```

# Allocating and Starting Jobs Interactively

- Get resources allocated

```
$ salloc -p dgx --ntasks=8 --gres=gpu:8
```

resource queue  
start n tasks in the machine per node (one per GPU required by some software framework)

Indicate that access to 8 GPU resources is required

- Submit start job in the allocated resources

```
$ srun --pty bash
```

```
$ srun hostname
```

# Allocating and Starting Jobs in Batch Mode

- Batch jobs are the preferred way of using the LRZ AI Systems.
- The **sbatch** command submits jobs described in a *sbatch script* file.
- Two additional arguments required in sbatch scripts: *output* and *error messages* file.
- After the preamble, the job to be executed is described.

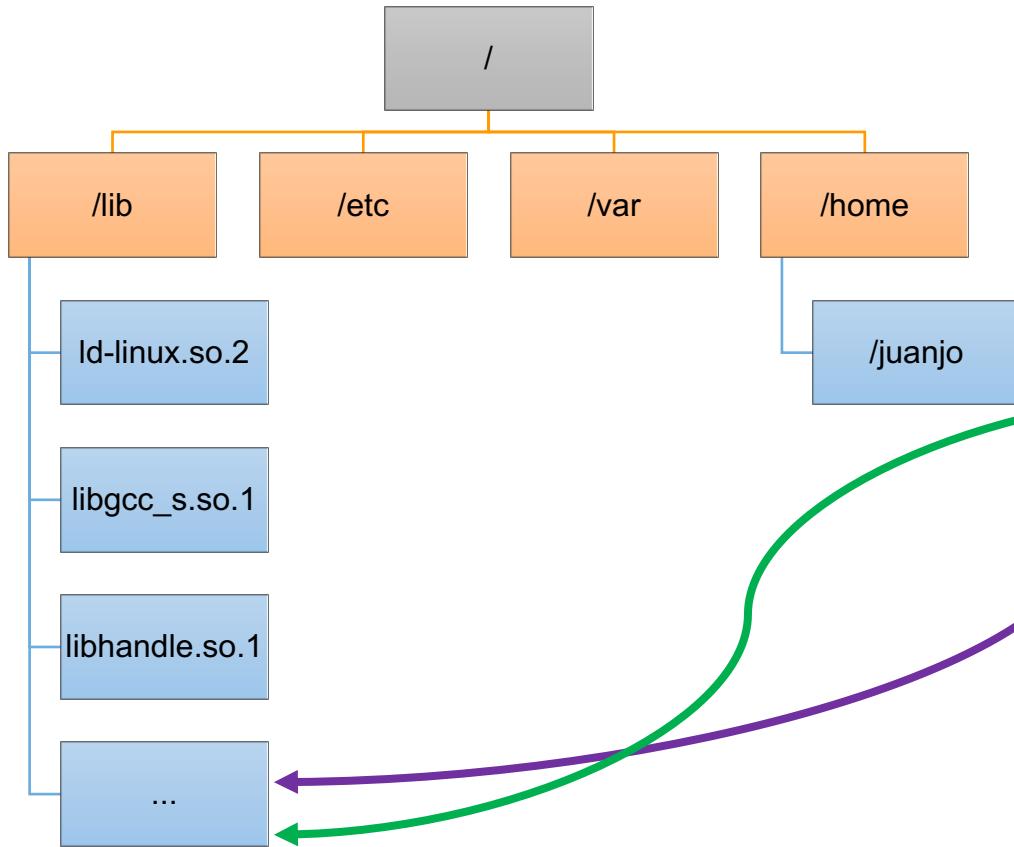
```
#!/bin/bash  
#SBATCH -N 1  
#SBATCH -p dgx-1-p100  
#SBATCH --gres=gpu:2  
#SBATCH --ntasks=2
```

```
srun nvidia-smi
```

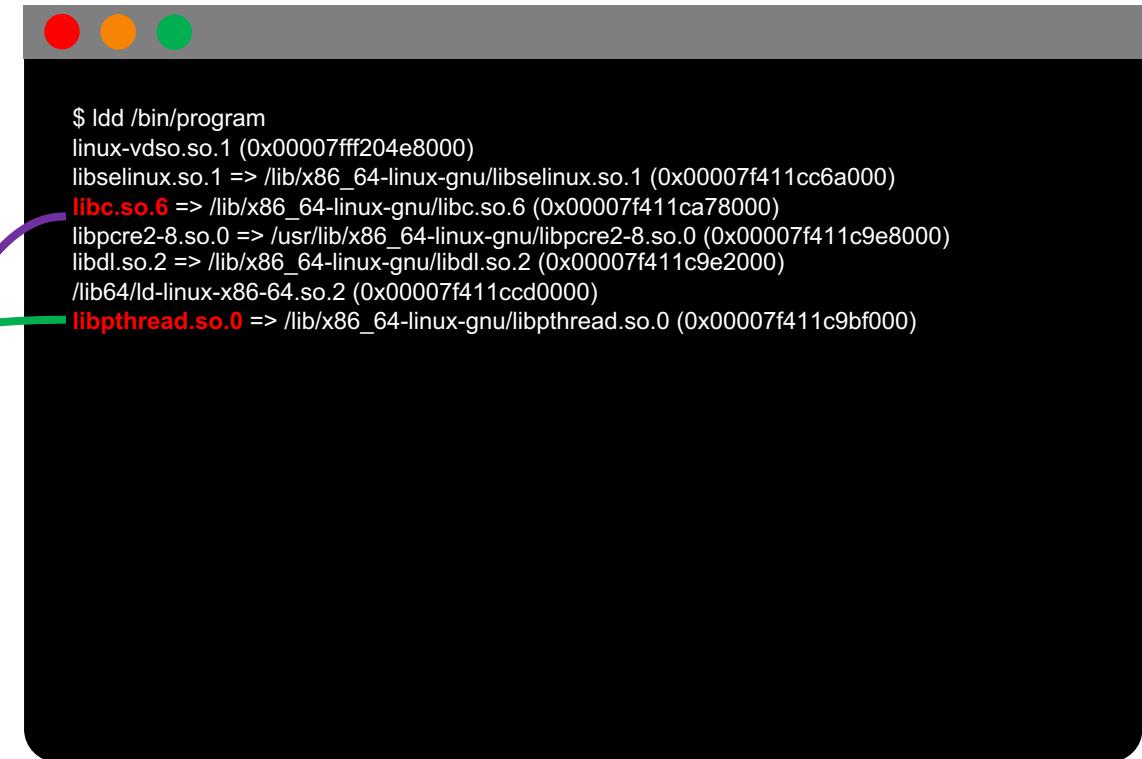
```
$ sbatch test.sbatch
```

# User Defined Software Stack: Container Technologies

## Typical Linux File System



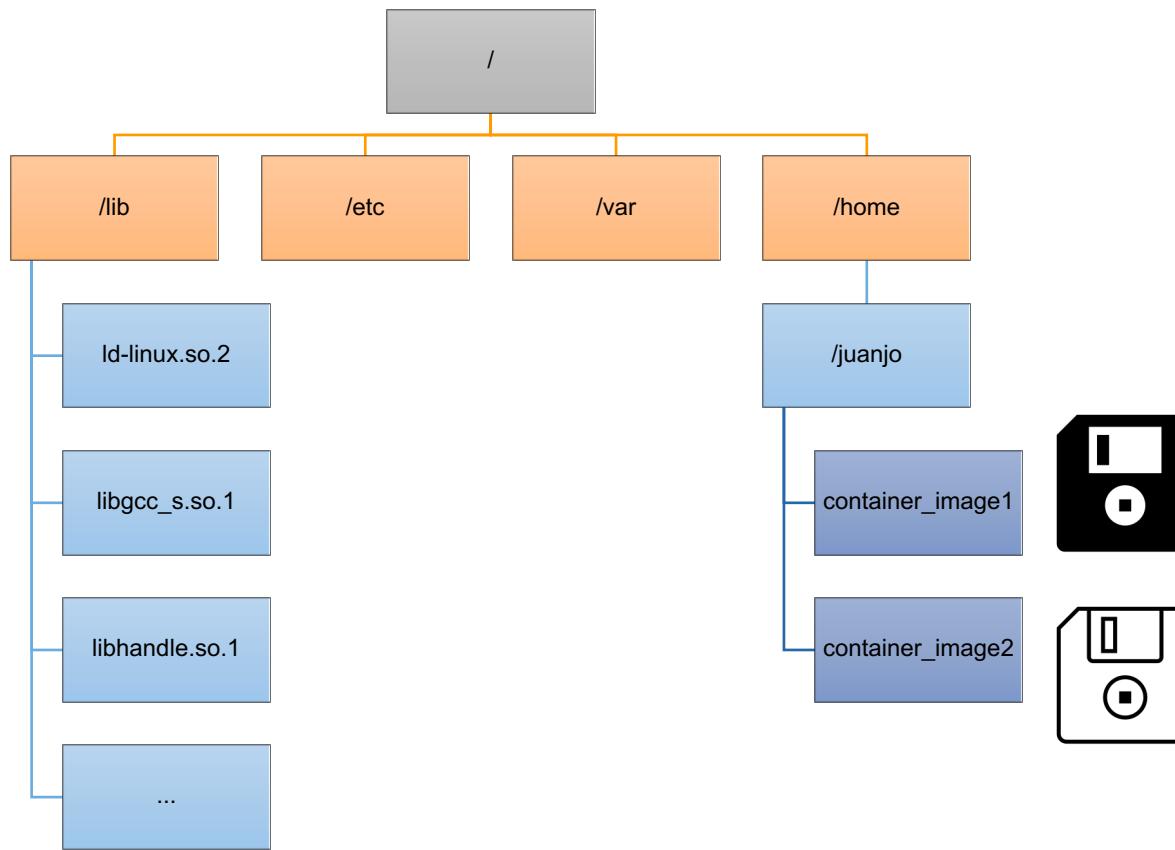
## Program In Execution in Linux



```
$ ldd /bin/program
linux-vdso.so.1 (0x00007fff204e8000)
libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so.1 (0x00007f411cc6a000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f411ca78000)
libpcre2-8.so.0 => /usr/lib/x86_64-linux-gnu/libpcre2-8.so.0 (0x00007f411c9e8000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f411c9e2000)
/lib64/ld-linux-x86-64.so.2 (0x00007f411ccd0000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f411c9bf000)
```

# User Defined Software Stack: Container Images

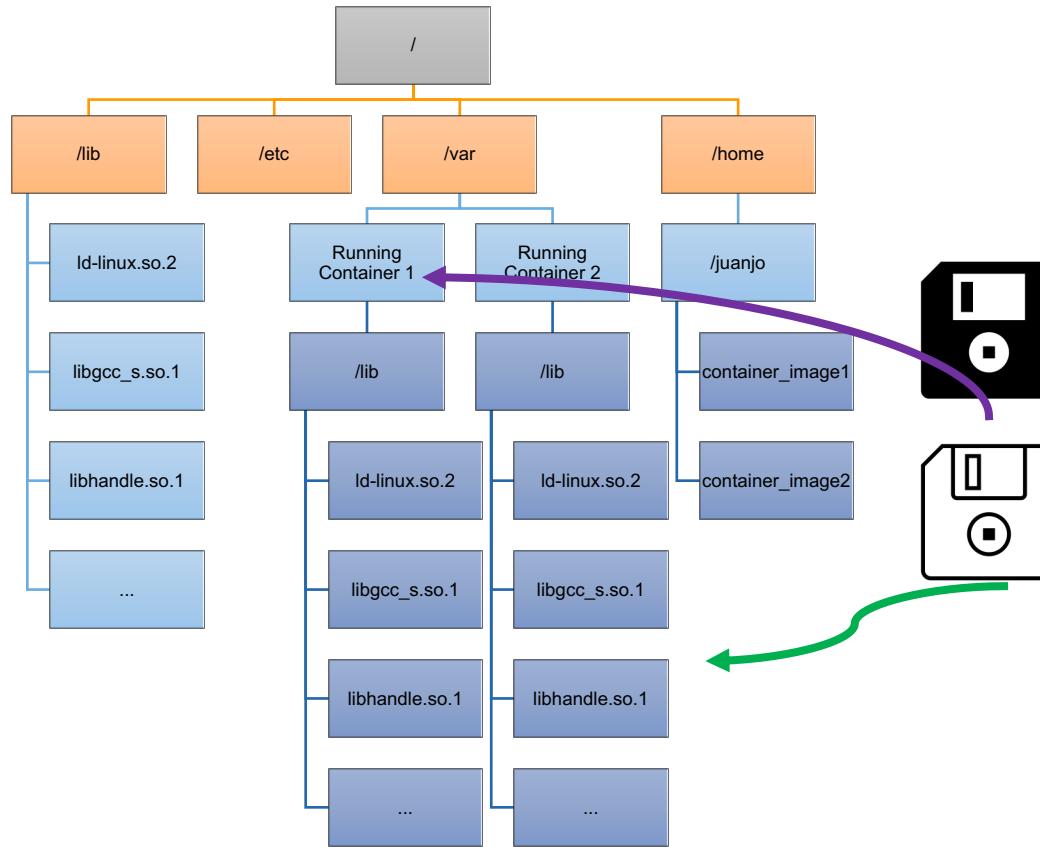
## Typical Linux File System



- Typically, a single compressed file
  - It contains a complete Linux File System + Metadata
- Different container technologies might:
  - use different formats
    - e.g., OCI format is a **specification for container images based on the Docker Image Manifest Version 2, Schema 2 format**
  - hide images to users
  - Are meant to be static
  - Not to be confused with a docker file

# User Defined Software Stack: Container

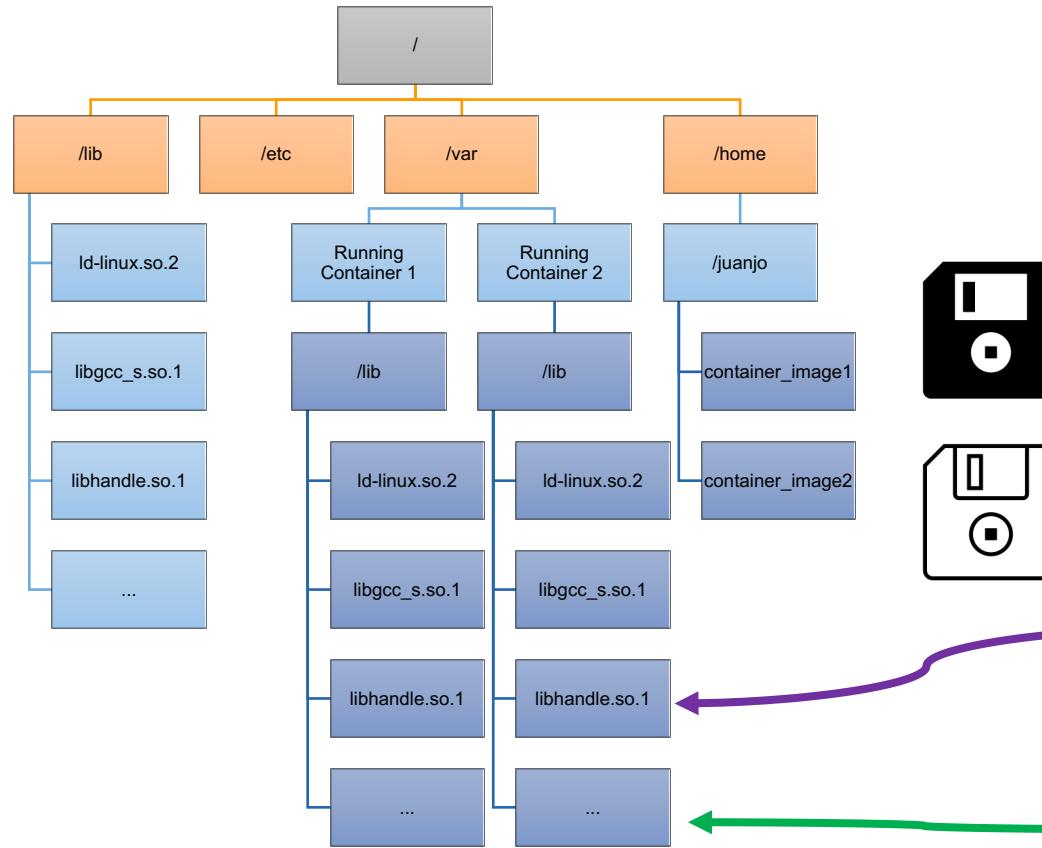
## Typical Linux File System



- A running instance of a container image
  - A complete Linux File System within a Linux File System
  - Libraries might be different (versions)
  - Provided programs might be different
- Specific program in charge of unpacking the image and storing it within the proper folder
  - Docker, Podman, Enroot, etc.
- More than one container can
  - Exist at any point in time
  - Be generated from a single image

# Understanding Containers: Container

## Typical Linux File System



- It is possible to “run a process within a container”
  - Confining the process to the content of the container File System
  - Specific program in charge for confining and running the process within the container
  - `docker run/start`, `enroot start`

```
$ ldd /bin/program
linux-vdso.so.1 (0x00007fff204e8000)
libsdl.so.1 => /lib/x86_64-linux-gnu/libsdl.so.1
(0x00007f411cc6a000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
(0x00007f411ca78000)
libpcre2-8.so.0 => /usr/lib/x86_64-linux-gnu/libpcre2-8.so.0
(0x00007f411c9e8000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2
(0x00007f411c9e2000)
/lib64/ld-linux-x86-64.so.2 (0x00007f411ccd0000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0
(0x00007f411c9bf000)
```

Process run within Running Container 2

# The Enroot Container Technology

- Containerized applications with enroot, a rootless container runtime by Nvidia
- Slightly different workflow than with Docker

```
#1 Import a container from a valid repository
```

```
$ enroot import docker://ubuntu
```

```
#2 Create an enroot container
```

```
$ enroot create ubuntu.sqsh
```

```
#3 Start/run an application within the container
```

```
$ enroot start ubuntu
```

- It should be noticed than the workflow in the AI System consist in submitting jobs that run containerized within an enroot defined container

# Nvidia NGC: Container Images Repository

**CATALOG**

**CONTAINERS**

Query: tensorflow

containers tab

**TensorFlow** Accelerated with NVIDIA

TensorFlow Container

TensorFlow is an open-source software library for high-performance numerical computation. Its flexible architecture allows easy deployment of computation a...

**NVIDIA L4T TensorFlow Container**

NVIDIA L4T TensorFlow Container

TensorFlow is an open-source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, wh...

**NVIDIA CLARA**

Model Analyzer Container

Model Analyzer, used to gather compute requirements for Triton inference models.

**NVIDIA JETSON**

NVIDIA L4T ML Container

The Machine learning container contains TensorFlow, PyTorch, JupyterLab, and other popular ML and data science frameworks such as scikit-learn, scipy, a...

**View Labels** **Pull Tag**

**View Labels** **Pull Tag**

**View Labels** **Pull Tag**

**View Labels** **Pull Tag**

**CATALOG**

**PRIVATE REGISTRY**

Catalog: Containers / Containers: nvidia:tensorflow

**TensorFlow**

Publisher	Built By	Latest Tag	Modified	Size
Google Brain Team	NVIDIA	20.12-tf1-py3	December 18, 2020	5.88 GB

**Multinode Support** Yes    **Multi-Arch Support** No

**Description**

TensorFlow is an open-source software library for high-performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

**Labels**

Deep Learning Training

**Pull Command**

```
docker pull nvcr.io/nvidia/tensorflow:20.12-tf1-py3
```

**Setup**

**Generate API Key**

**My Account Settings**

**Setup**

**Terms of Use**

**Privacy Policy**

**Sign Out**

**Setup > API Key**

**API**

**API Information**

Generate your own API key in order to use the NGC service through the Docker client or through NGC CLI.

**Usage**

Use your API key to log in to the NGC registry by entering the following command and following the prompts:

**NGC CLI**

```
$ ngc config set
```

**Docker™**

For the username, enter '\$oauthtoken' exactly as shown. It is a special authentication token for all users.

```
$ docker login nvcr.io
Username: $oauthtoken
Password: <your Key>
```

**Docker™**

For the username, enter '\$oauthtoken' exactly as shown. It is a special authentication token for all users.

```
$ docker login nvcr.io
Username: $oauthtoken
Password: OHBodjU6NGE4NwM1NzUtZDE1ZS0t
```

**API Key generated successfully. This is the only time your API Key will be displayed. Keep your API Key secret.**

**Do not share it or store it in a place where others can see or copy it.**

**API Key:** HBodjU6NGE4NwM1NzUtZDE1ZS0t

**Key (partially shown here for security reasons)**

# Beyond Existing NGC: Creating Custom Enroot Image

- Get resources / get an allocation of resources
  - `salloc -p test-v100x2 -q testing --gres=gpu:1`
- Open a terminal on the allocated resources
  - `srun --pty bash`
- (Optional) Create a base container image (e.g., pulling from NGC)
  - `enroot import -o pytorch_base.sqsh docker://nvcr.io#nvidia/pytorch:22.06-py3`
- Create a container out of an image
  - `enroot create --name pytorch_container pytorch_base.sqsh`

# Beyond Existing NGC: Creating Custom Enroot Image

- Start a terminal your recently created container
  - `enroot start pytorch_container bash`
- Modify your container accordingly
- Leave your container
- Export the created container as an image
  - `enroot export --output hugging_face.sqsh pytorch_container`

# Using the Cluster with Enroot Containers

- Get resources allocated

```
$ salloc -p dgx --ntasks=8 --gres=gpu:8
```

Indicate that access to the GPU resources is required

- Submit containerized job

```
$ srun --pty enroot start --mount ./data-test:/mnt/data-test ubuntu bash
```

mounting outside folders inside the container

the container previously created

- Meet the pyxis plugin: container creating and job submission in a single step

```
$ srun --container-mounts=./data-test:/mnt/data-test --container-name=horovod --container-image='horovod/horovod+0.16.4-tf1.12.0-torch1.1.0-mxnet1.4.1-py3.5' bash
```

# Open On Demand: Web Frontend for the LRZ AI System

- Interactive web service for AI systems where Jupyter Notebook, JupyterLab and RStudio Server environments are available, at <https://datalab3.srv.lrz.de>.
- Given that requested resources are available, the status of the session will change from "Queued" to "Starting" and finally "Running".
- <https://doku.lrz.de/display/PUBLIC/LRZ+AI+Systems>

The screenshot shows the LRZ AI Systems Web UI interface. At the top, there's a navigation bar with links for Home, My Interactive Sessions, and Jupyter Notebook. Below the navigation, a sidebar lists 'Interactive Apps' and 'Servers', with 'Jupyter Notebook' selected and highlighted in blue. The main content area is titled 'Jupyter Notebook version: 0d9944a' and 'Jupyter Notebook Access'. It includes dropdown menus for selecting a partition ('gpu-v100') and an Nvidia NGC container image ('Pytorch'). There are input fields for 'Number of hours' (set to 1) and 'Desired number of GPUs for your job' (set to 1). A text field for 'Comma separated list of mounts' is present, though empty. A checkbox labeled 'Make it Jupyter Lab!' is checked. At the bottom right is a large blue 'Launch' button.

# The Compute Cloud

# Cloud Computing Characteristics

A consumer can request and receive access to a service offering without an administrator or some sort of support staff having to fulfil the request manually.

on demand

Cloud services should be easy to access. Ideally only a basic network connection should be required.

broad  
network  
access

Ability to grow with user demand. If the system is well defined it should be relatively easy for the provider to add more

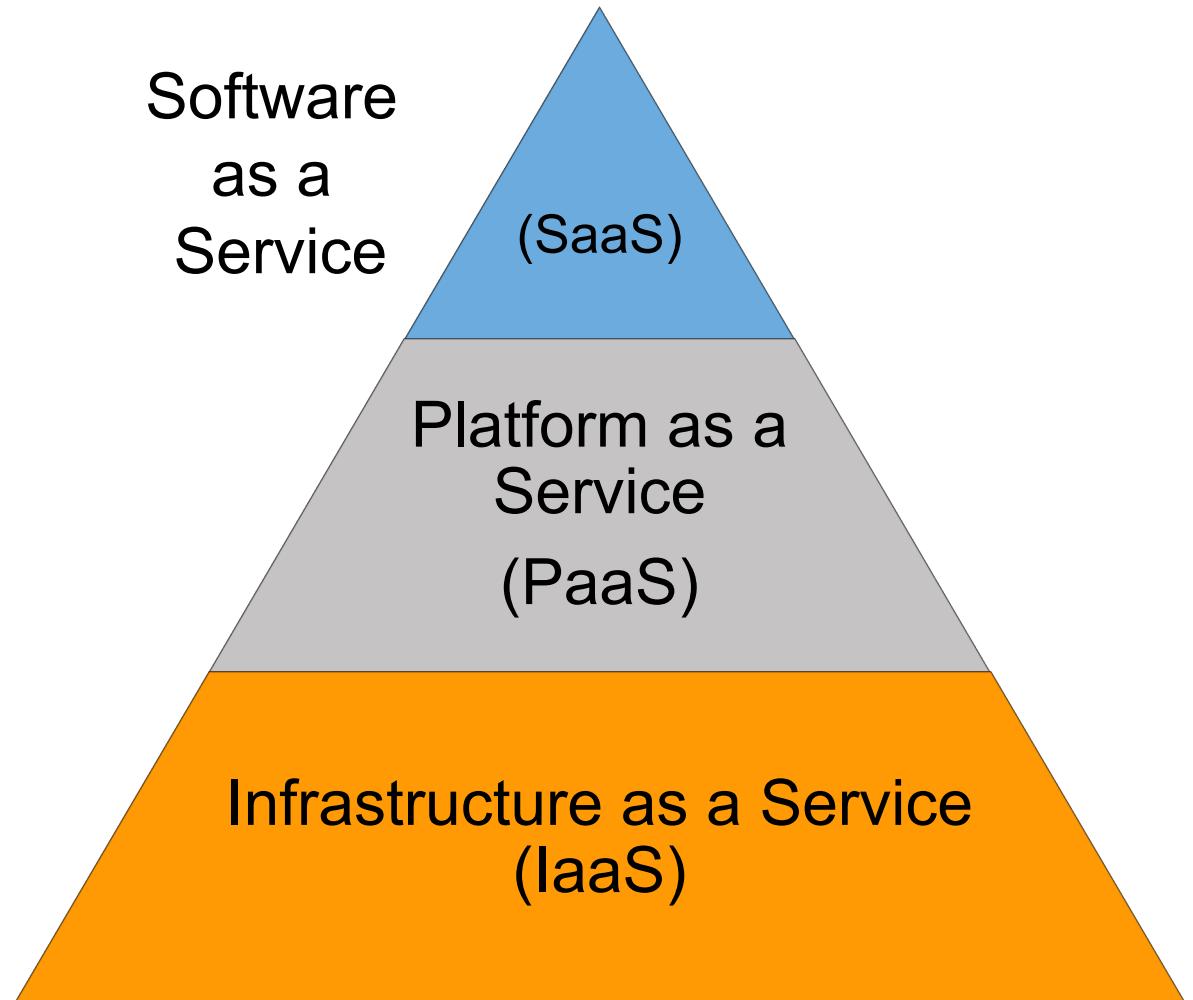
flexible

Cloud services must have the ability to measure usage. Usage can be quantified using various metrics, such as time, bandwidth used, and data used. This ability to measure allows what is known as pay as you go model.

measurable

A user will not need all the resources available to her. When resources are not used, they should be released, and other users can benefit from them or they can be simply not used (not consuming energy.)

resource  
pooling



- SaaS – Fully developed software solution to be used
  - e.g., Google Drive
- PaaS – Provides a framework on top of which is possible to build, deploy, and manage software products
  - e.g., Heroku
- IaaS - Provides a completely virtualized computing infrastructure provisioned and managed over the internet
  - e.g., LRZ Compute Cloud

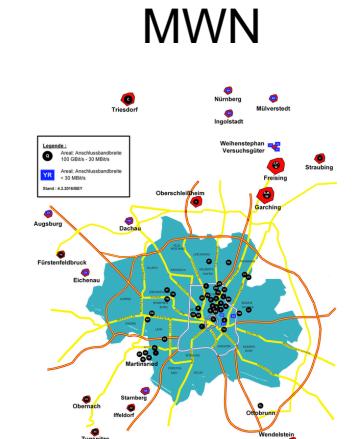
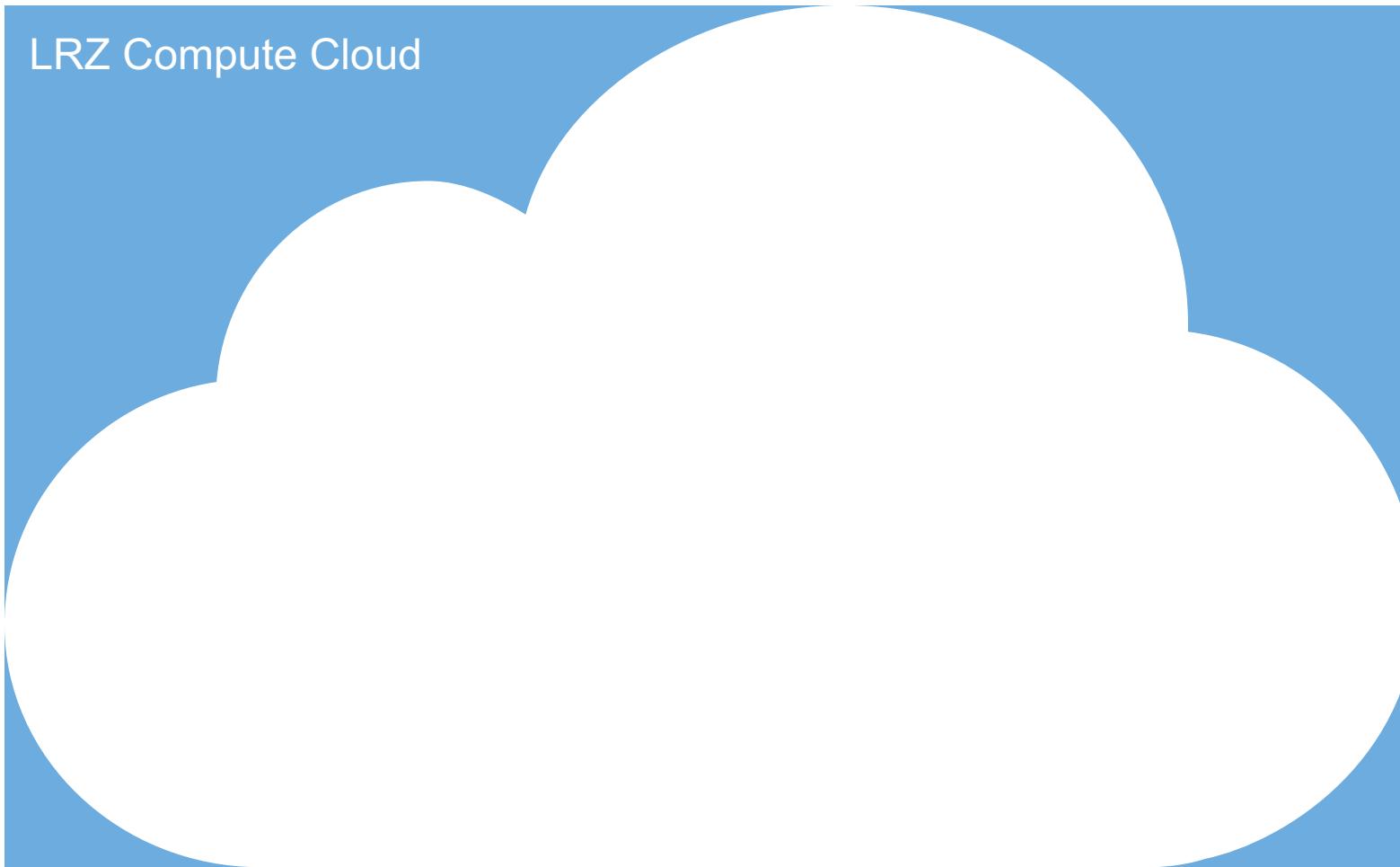
# OpenStack: The Engine of the LRZ Compute Cloud

- What do we need for transforming a set of resources (data center) into a cloud?
  - to manage/admin the hardware
  - to provision machines to users
  - to allow users to authenticate
  - to manage the network across resources
  - ...

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed and provisioned through APIs with common

- OpenStack bundles together a bunch of different technologies, addressing the different needs transforming resources into a Cloud Service

# The LRZ Compute Cloud at a Glance



# OpenStack - Terminology

- **Image**

A single file which contains a virtual disk with a bootable operating system installed on it. Images are like a template of a computer's root drive. They contain the operating system and can also include software and layers of your application, such as database servers, web servers, and so on.



FreeBSD



# OpenStack - Terminology

- **Instance**

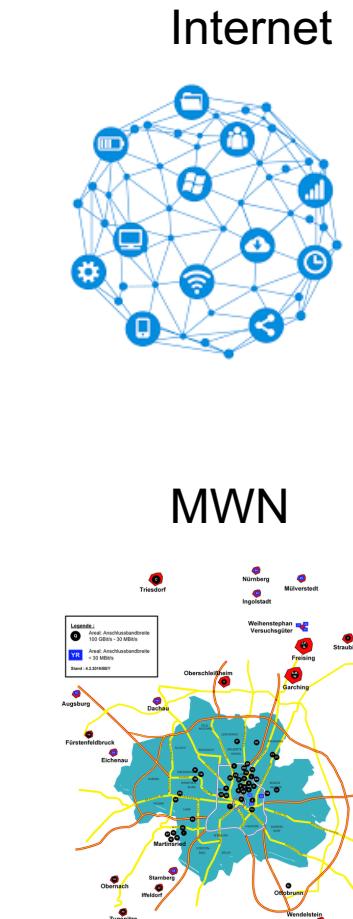
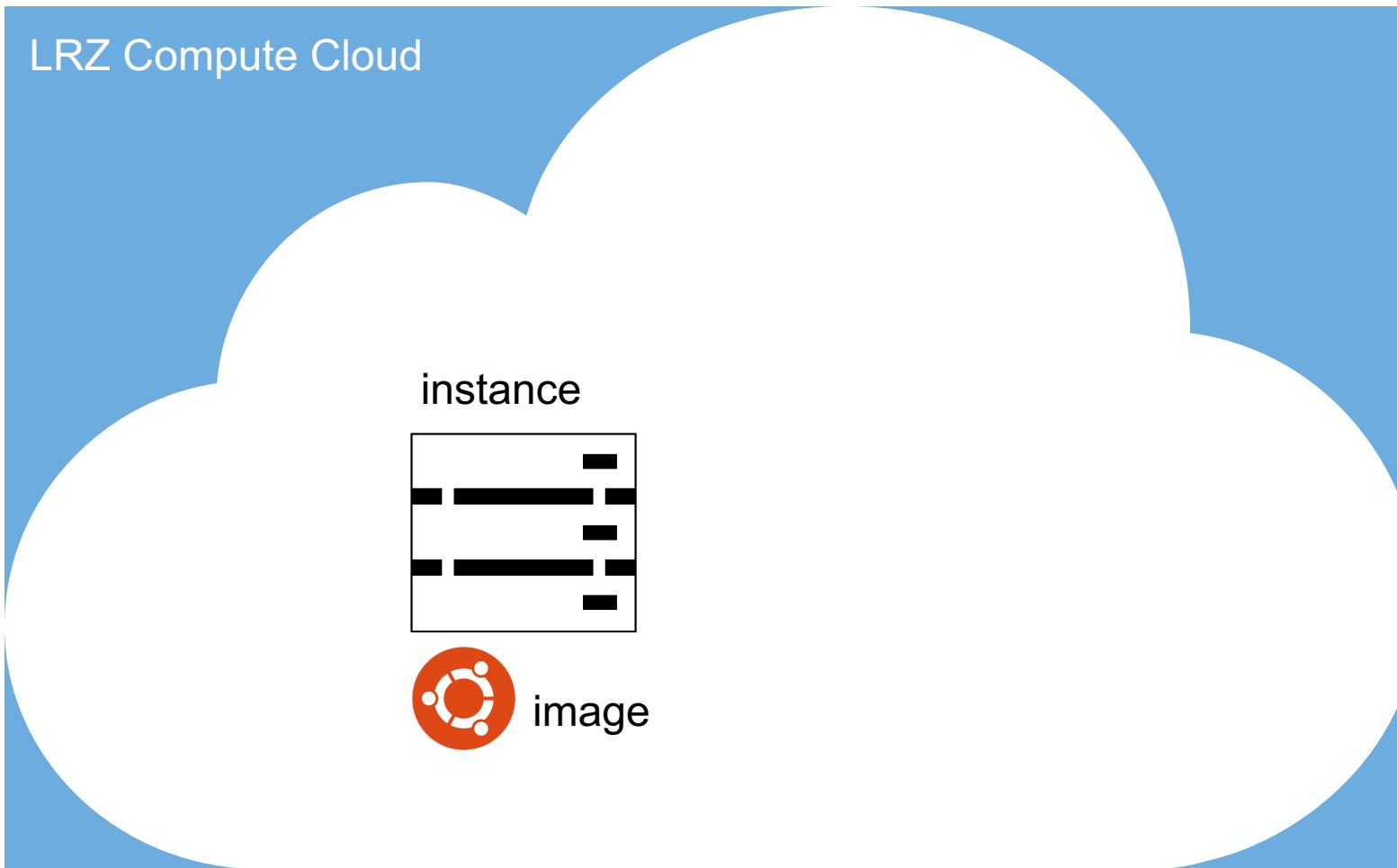
A copy of an image running as a virtual server in the cloud. We will also call it server.

- **Flavor**

Flavors define the compute, memory, and storage capacity of instances. To put it simply, a flavor is an available hardware configuration for a server.

Name	vCPUs	RAM	Remarks	Access
tiny	1	512 MB	for testing purposes only, most Operating Systems will not boot due to restricted resources	public
nvidia-v100.2	40	700 GiB	use 2 GPUs on a GPU node (use entire GPU node)	restricted, contact us
nvidia-v100.1	20	350 GiB	use 1 GPU on a GPU node	restricted, contact us
lrz.xlarge	10	47.5 GiB	use 1/4 compute node	public
lrz.xhuge	48	1488 GiB	use 1/4 of the hugemem node	restricted, contact us
lrz.small	1	4.75 GiB	use 1/40 compute node	public
lrz.medium	2	9.5 GiB	use 1/20 compute node	public
lrz.large	4	19 GiB	use 1/10 compute node	public
lrz.huge	24	744 GiB	use 1/8 of the hugemem node	restricted, contact us
lrz.4xlarge	40	190 GiB	use entire compute node	restricted, contact us
lrz.4xhuge	192	5952 GiB	use entire hugemem node	restricted, contact us
lrz.2xlarge	20	95 GiB	use 1/2 compute node	restricted, contact us
lrz.2xhuge	96	2976 GiB	use 1/2 of the hugemem node	restricted, contact us

# The LRZ Compute Cloud at a Glance

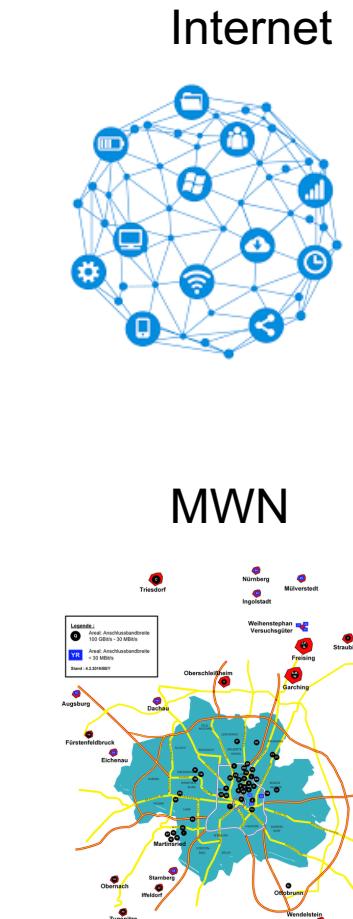
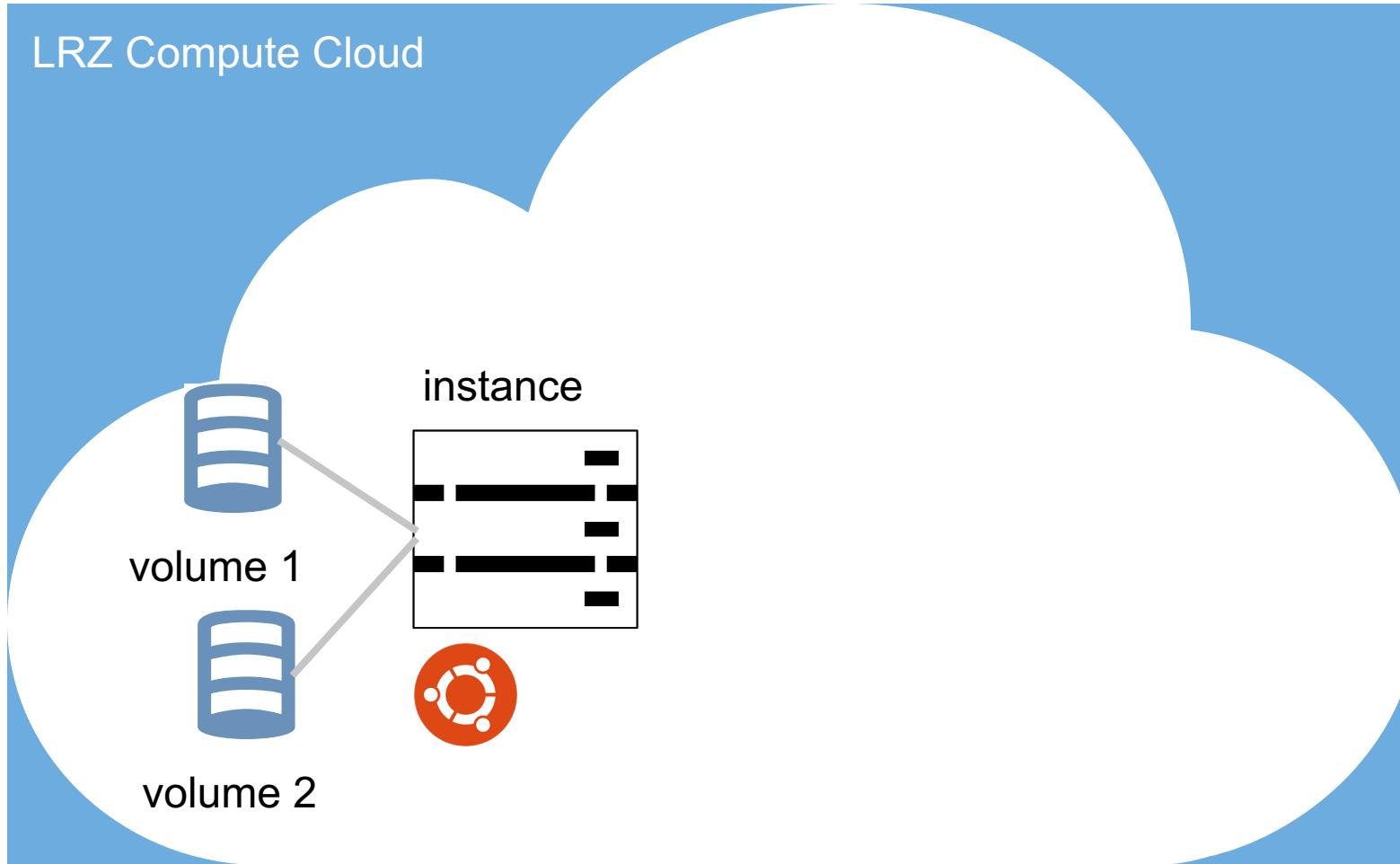


# OpenStack - Terminology

- **Volume**

A volume is a detachable block storage device, similar to a USB hard drive. You can attach a volume to only one instance. But an instance can attach several volumes

# The LRZ Compute Cloud at a Glance



# OpenStack - Terminology

- **Networking**

OpenStack provides networks, subnets, and routers as object abstractions. Each abstraction has functionality that mimics its physical counterpart: networks contain subnets, and routers route traffic between different subnets and networks. Instances are created within internal private networks. These networks can be routed to external networks (e.g., Internet or MWN) via a virtual router.

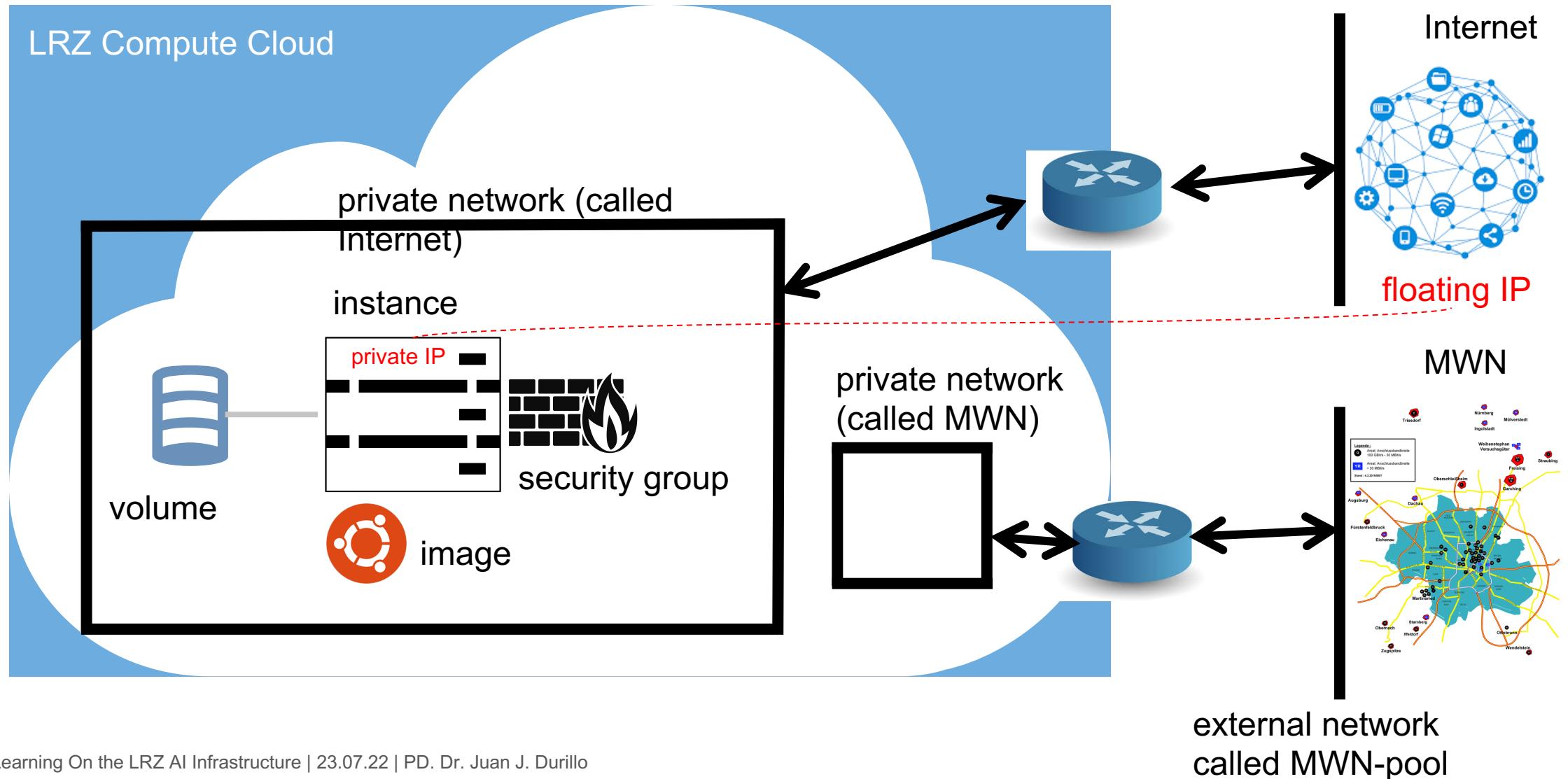
- **Private and Floating IP**

Each instance has a fixed IP within its private Network. That IP can be associated to an IP of the external network that network is connected by means of what it is called *floating IP address*. The floating IP address will allow addressing the instance from the outside.

- **Security group**

A security group acts as a virtual firewall for servers and other resources on a network. It is a container for rules for allowing different types of network traffic to and from an instance.

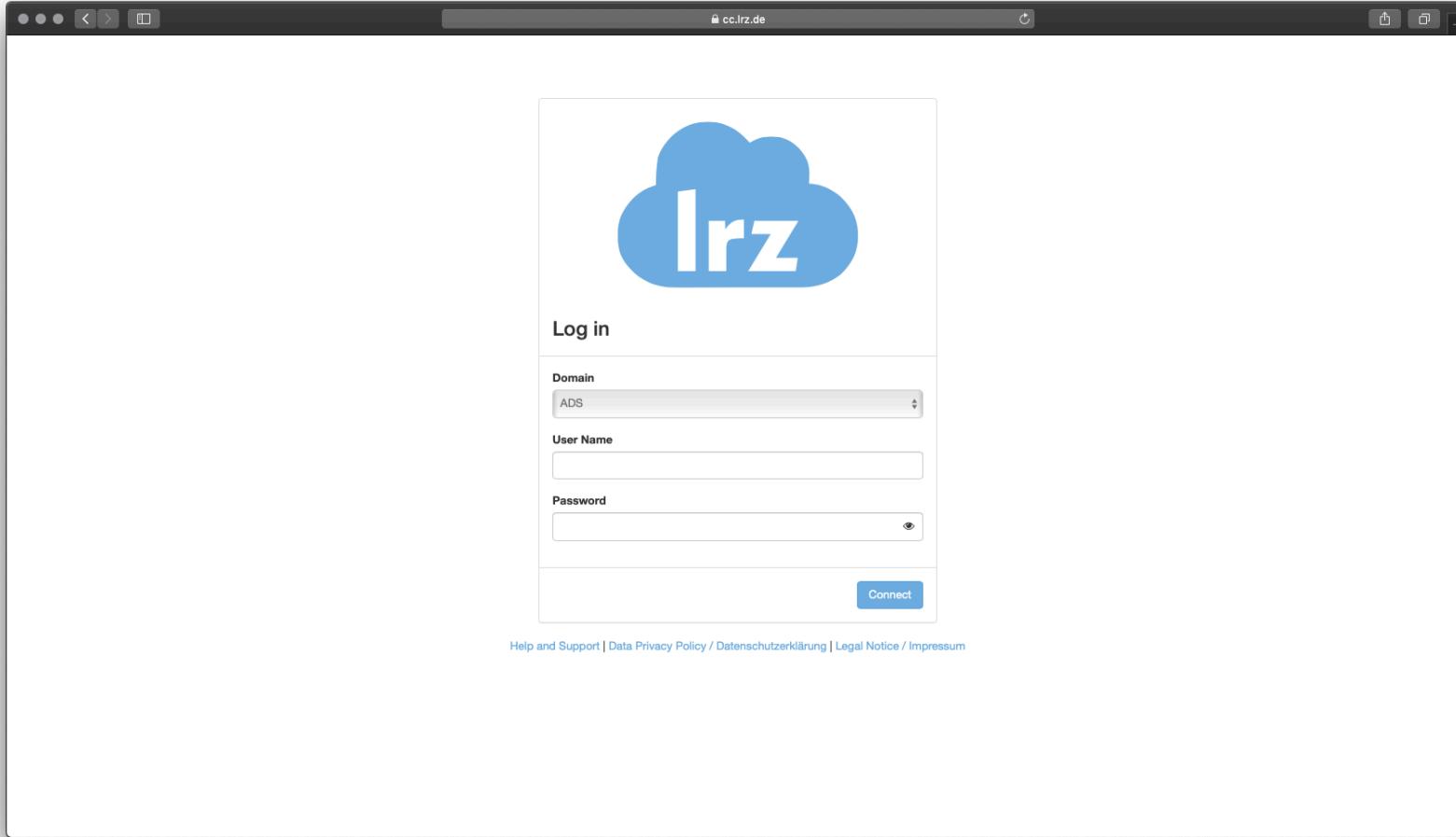
# The LRZ Compute Cloud at a Glance



# A server on the Compute Cloud

- De-facto operation is no Graphical Interface on the provided images
  - Although it is possible (e.g., <https://rv.lrz.de> ← only accessible from within MWN)
- Accessing instances via ssh
  - No login based on password by default (public and private keys!)
- OpenStack must be aware of your public key(s) to add it(them) to newly created instances (otherwise you will not be able to login)
  - You can import a public key of a keypair generated using your method of preference
  - You can generate a keypair using OpenStack
    - the private key will be downloaded to your computer
    - the public will be recorded by OpenStack

# The Compute Cloud Web Interface



# The Compute Cloud Web Interface

Left panel.  
Allows you to  
operate with the CC.

The screenshot shows the LRZ Compute Cloud web interface. The left sidebar is highlighted with an orange box and contains the following navigation items under the 'Compute' section:

- Overview (selected)
- Instances
- Images
- Key Pairs
- Volumes >
- Network >
- Orchestration >
- Identity >

The main content area is titled 'Overview' and includes a 'Limit Summary' section with six pie charts showing usage against limits:

- Instances: Used 1 of 4
- VCpus: Used 40 of 160
- RAM: Used 753,664 (No Limit)
- Floating IPs: Allocated 6 of 50
- Security Groups: Used 8 of 10
- Volumes: Used 5 of 10

Below this is a 'Usage Summary' section with a date range selector from 2019-09-29 to 2019-09-30. The summary table shows:

Active Instances:	1
Active RAM:	736GB
This Period's VCPU-Hours:	1504.24
This Period's GB-Hours:	752.12
This Period's RAM-Hours:	28342240.03

The 'Usage' section displays one item:

Instance Name	VCpus	Disk	RAM	Time since created
demo-isc	40	20GB	736GB	4 months, 3 weeks

Resources Limits.  
Shows you what  
you are allow and  
what you have in  
use.

Summary.  
Show the  
instances you  
have. They might  
be in different  
status.

# The Compute Cloud Web Interface

Allows a more informative view of our instances

The screenshot shows the LRZ Compute Cloud Instances page. The left sidebar has 'Compute' selected under 'Project'. The main area shows a table with one row:

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
demo-lsc	vol:Ubuntu-18.04-LTS-bionic	192.168.128.43 Floating IPs: 138.246.232.37	nvidia-v100.2	lrz_durillo_key	Shelved Offloaded	nova	None	Shut Down	4 months, 3 weeks	<button>Disassociate Floating IP</button>

Annotations on the page:

- 'Instances' button in the sidebar (highlighted)
- 'Image used by this instance' (points to 'Image Name')
- 'key used to log in' (points to 'Key Pair')
- 'Allows creating new instances / Deleting existing ones' (points to 'Actions' button)
- 'Instance Name. Not mandatory, but convenient as will see later in this course' (points to 'Instance Name')
- 'Resources used by this instance' (points to 'IP Address' and 'Flavor')
- 'Status of the instance.' (points to 'Status')
- 'Summary of this instance IPs' (points to 'IP Address' column)

Allows creating new instances / Deleting existing ones

Allows operating this instance (e.g., attach a volume, associate new floating IP)

# The Compute Cloud Web Interface

Project / Compute / Key Pairs

Key Pairs

Name	Fingerprint	Action
access_from_windows	51:93:d4:5f:c1:0b:09:fd:ca:11:19:e3:1e:34:ec:32	Delete Key Pair
lrz_durillo_key	c8:aa:48:d1:64:03:42:b5:bb:b9:0:a4:4d:9f:41:a9	Delete Key Pair

Project / Volumes / Volumes

Volumes

Name	Description	Size	Status	Type	Attached To	Availability Zone	Bootable	Encrypted	Action
38960ca03-2f67-4417-91b0-6a7d32d35cc8	-	30GiB	In-use	ceph	/dev/vda on test	nova	Yes	No	Edit Volume
0c5b5550-947b-44c5-a20a-46b298a5d97	-	30GiB	Available	ceph	-	nova	Yes	No	Edit Volume
10a34d24-1d7b-46a4-bc3e-f1558a1d3918	-	30GiB	Available	ceph	-	nova	Yes	No	Edit Volume
tensorflow-gpu-volume	-	25GiB	Available	ceph	-	nova	Yes	No	Edit Volume
c706810c-dcd3-4fea-9214-aecd8ecfc901	-	20GiB	Available	ceph	-	nova	Yes	No	Edit Volume
63e9cb82-3bb5-44d3-be13-7421d12ff5a2	-	20GiB	Available	ceph	-	nova	Yes	No	Edit Volume
data		80GiB	Reserved	ceph	-	nova	No	No	Update Metadata
f5bbb8b3-ce0c-4c0f-a79a-b390ee8a72d0	data for isc 2019 demo	20GiB	Reserved	ceph	-	nova	Yes	No	Update Metadata

# A Guided Example

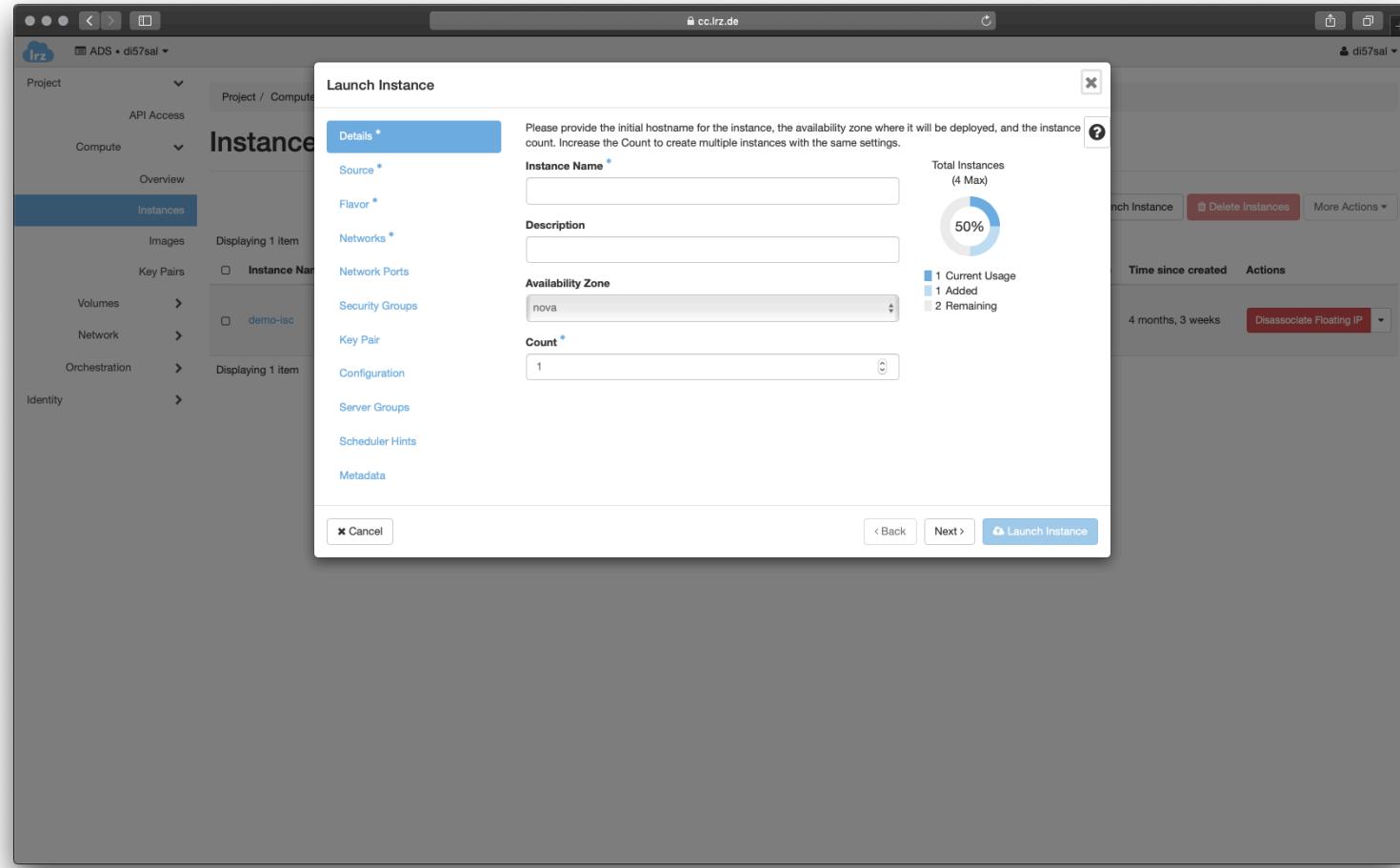
## TASK : Create a Ubuntu-based server for running Jupyter Notebooks

- On the Compute Cloud Web Interface
  - Generate a new keypair (follow along the live demo)
- On your computer
  - A file with the extension .pem will be downloaded to your machine (the private key) from previous step
  - In Linux/UNIX: change the permission of that file to 600 (`$ chmod 600 ...`)
  - In Windows with WLS: copy the downloaded file to inside the WSL (/mnt/c/ allows you accessing C:\ in windows from WSL,)
    - Once copied, change the permissions as in the Linux/UNIX case
  - In Windows with Putty: import it using PuttyGen
    - check <https://stackoverflow.com/questions/3190667/convert-pem-to-ppk-file-format> if you need help

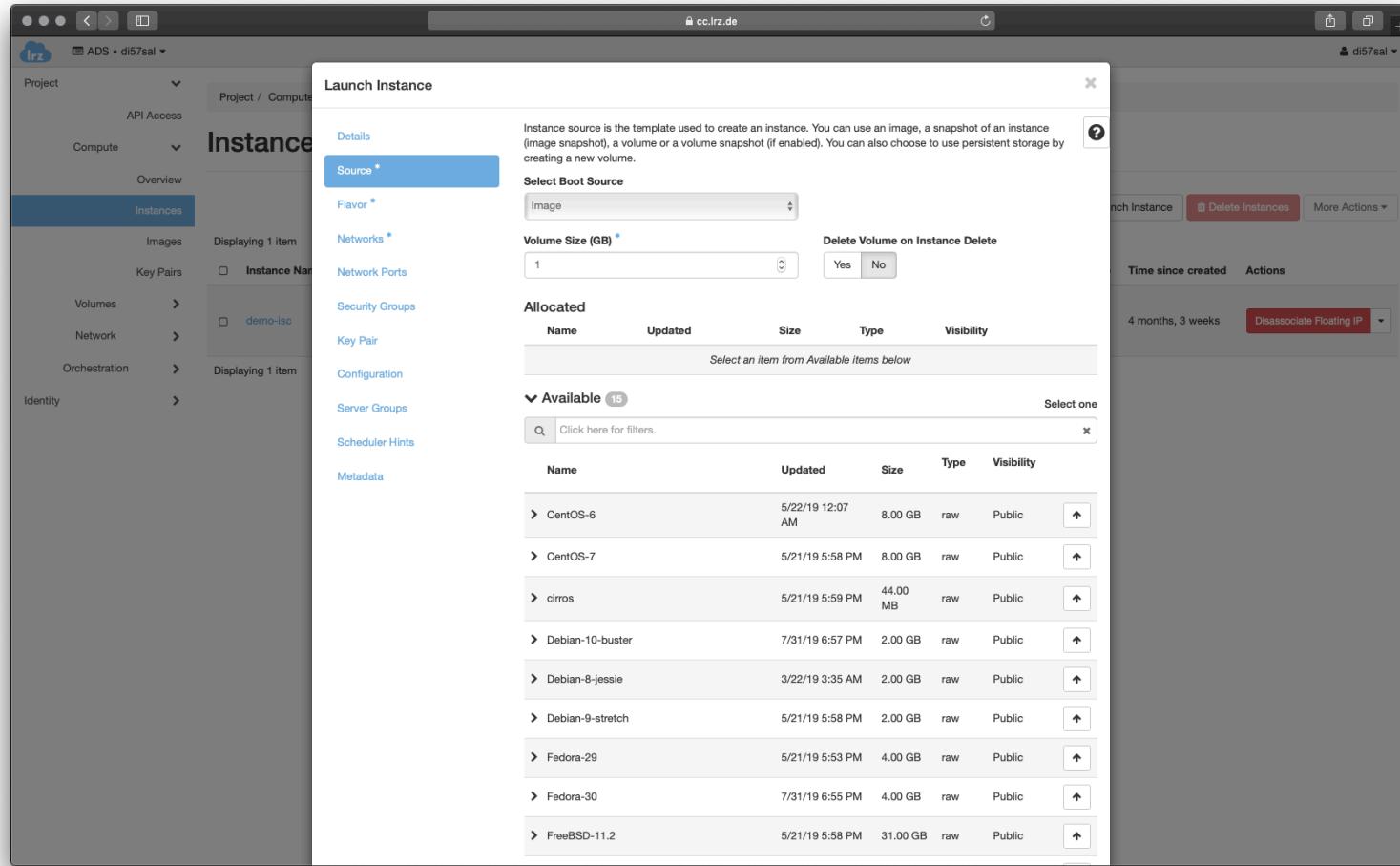
# A Guided Example

- [On the Compute Cloud Web Interface](#)
  - Create an instance (next steps are documented with screen shots in successive slides)
    - Choose Ubuntu as image
    - CPU only flavor (preferably a small one)
    - Should be accessible from Internet
      - Place the instance on the private network called internet
      - Once the instance is created assign it a floating IP from the Internet pool

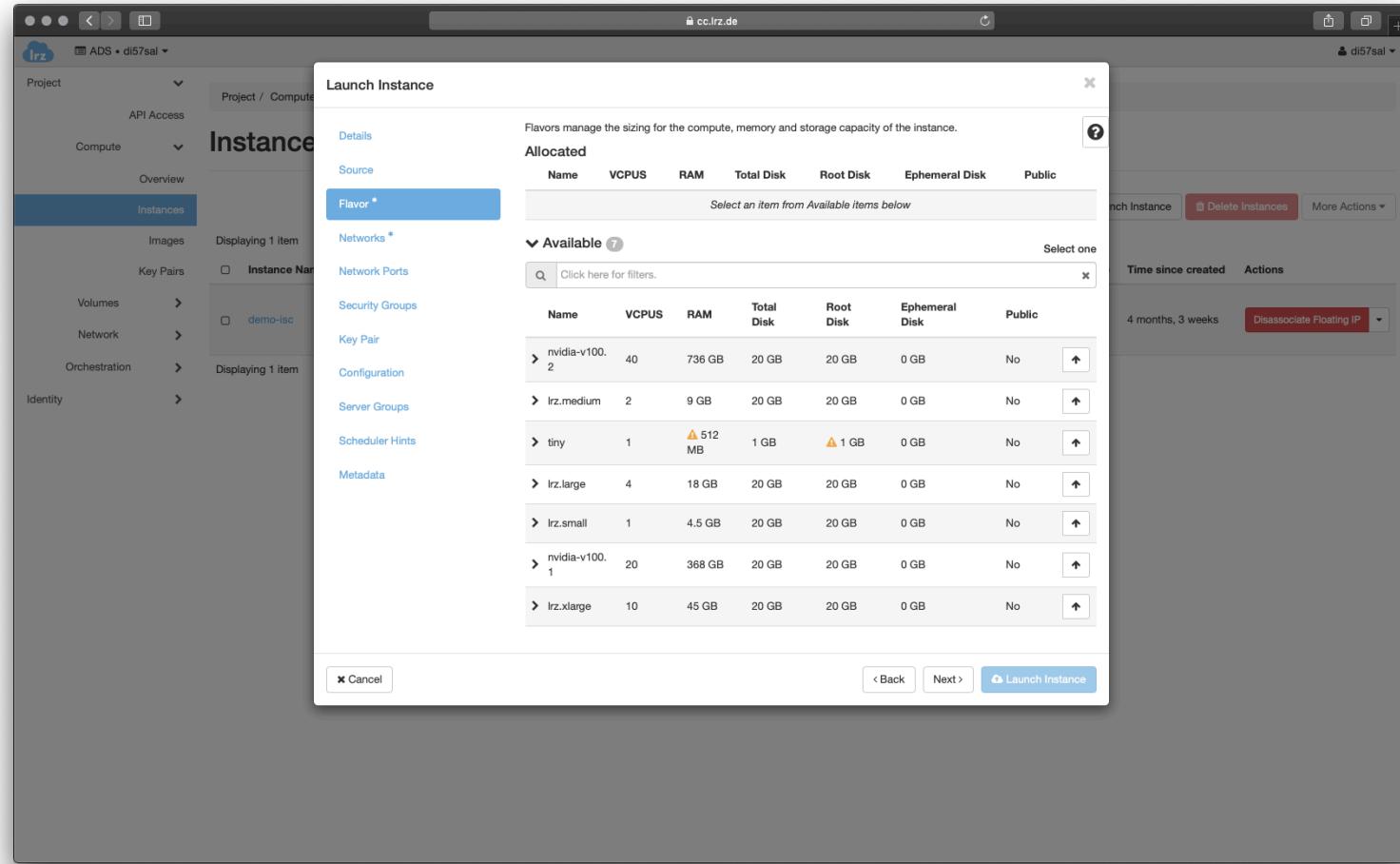
# Creating an Instance using the Web Interface



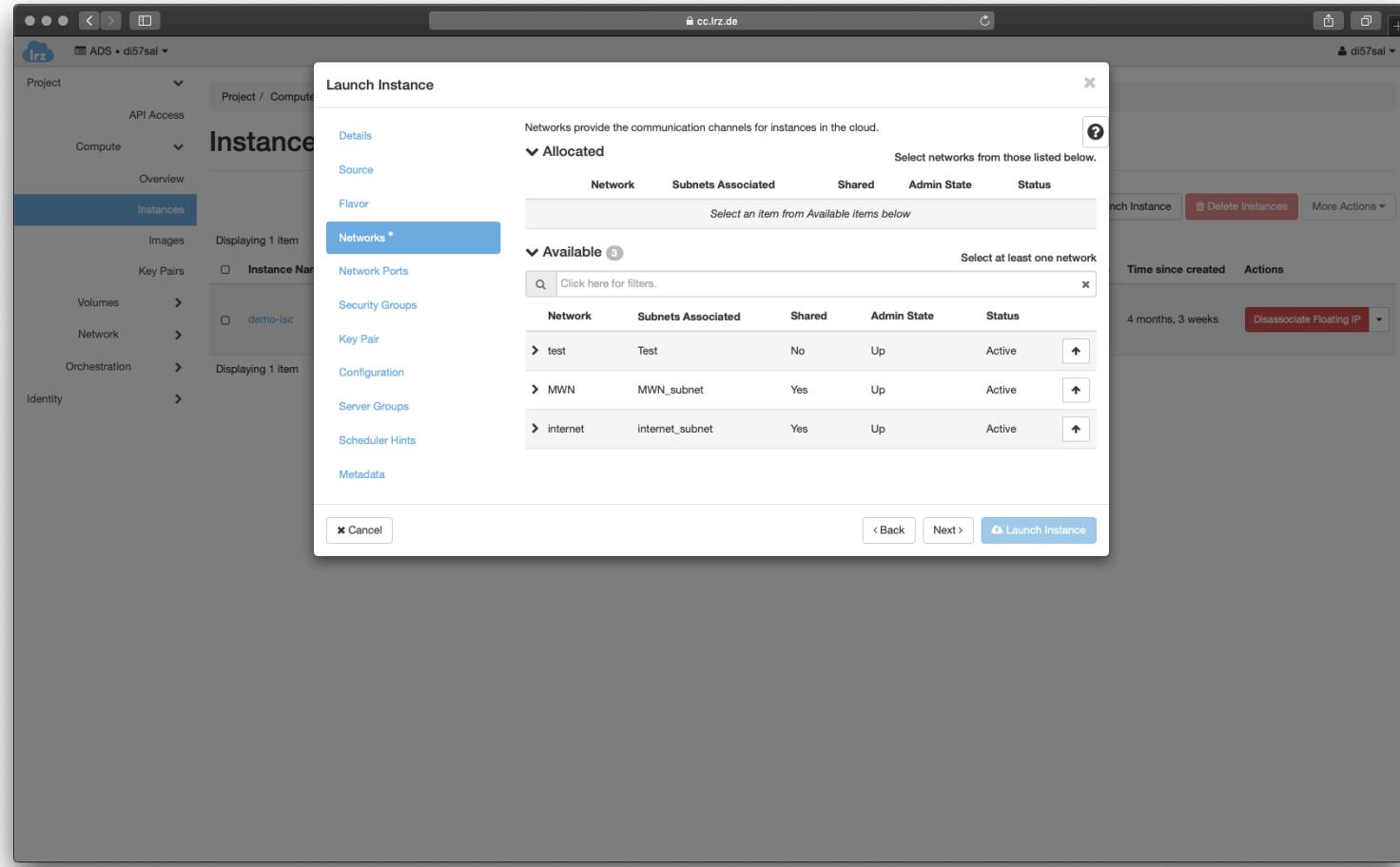
# Creating an Instance using the Web Interface



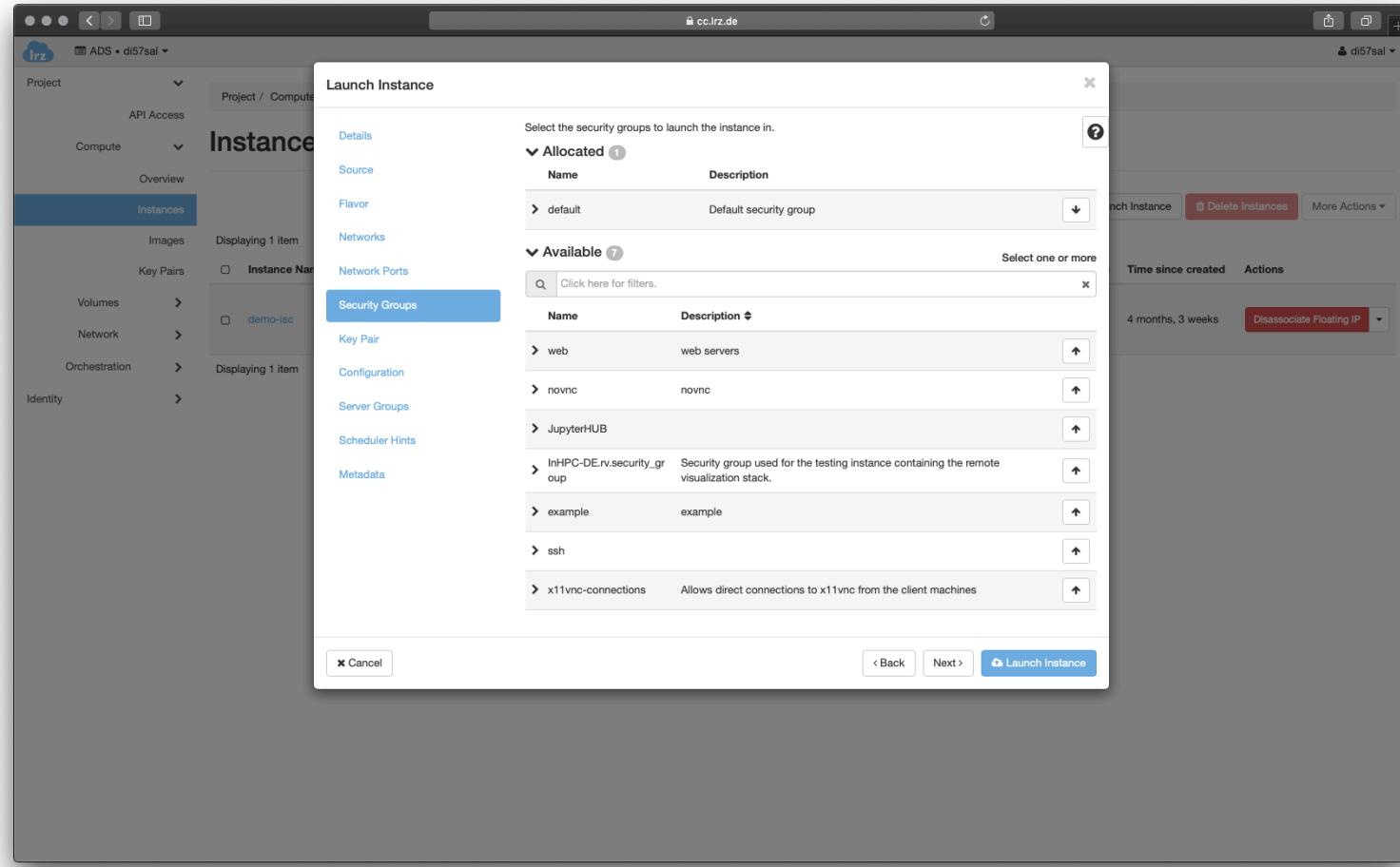
# Creating an Instance using the Web Interface



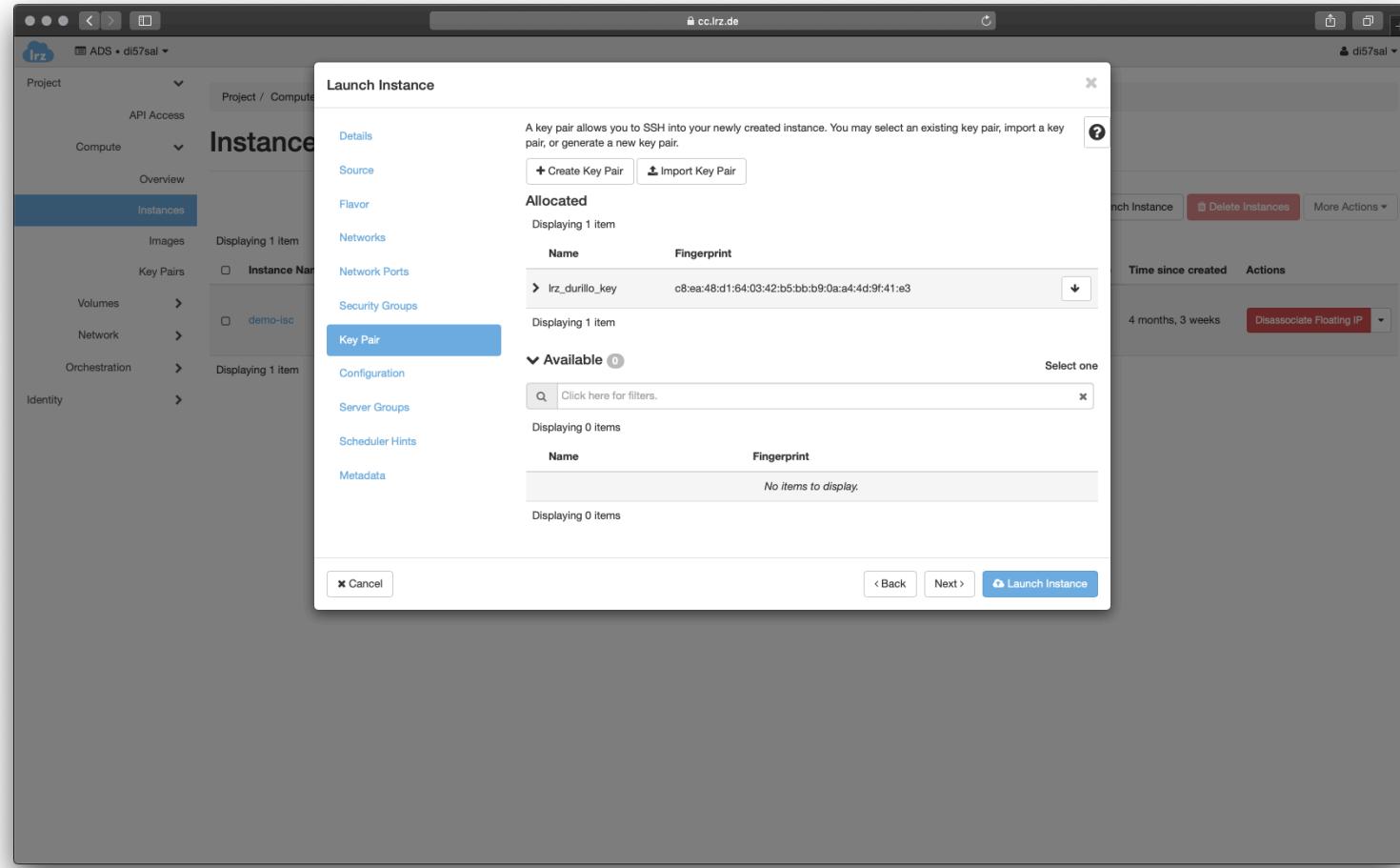
# Creating an Instance using the Web Interface



# Creating an Instance using the Web Interface



# Creating an Instance using the Web Interface



# Creating an Instance using the Web Interface

The screenshot shows the LRZ Compute Cloud web interface. The left sidebar has sections for Project, API Access, Compute (with Overview, Instances selected, Images, Key Pairs), Volumes, Network, Orchestration, and Identity. The main content area is titled 'Instances' and displays two items: 'test' and 'demo-isc'. The 'test' instance is active, running on nova, with an IP address of 192.168.128.119. The 'demo-isc' instance is offloaded, shelved, and shut down, with floating IPs 192.168.128.43 and 138.246.232.37. A context menu is open over the 'test' instance, listing actions like Associate Floating IP, Attach Interface, Detach Interface, Edit Instance, Attach Volume, Detach Volume, Update Metadata, Edit Security Groups, Console, View Log, Shelve Instance, Resize Instance, Soft Reboot Instance, Hard Reboot Instance, Shut Off Instance, and Delete Instance.

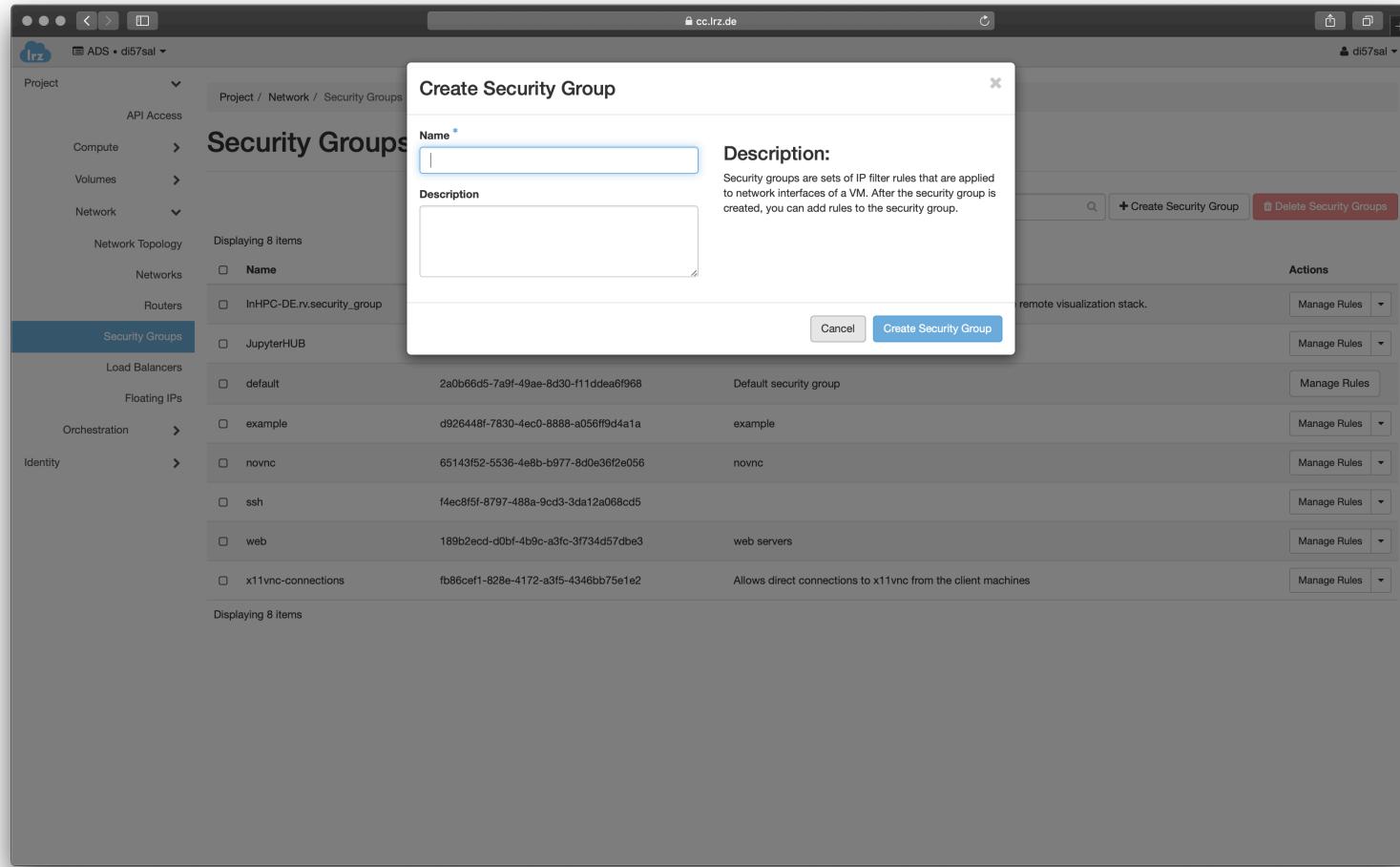
Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
test	vol:Ubuntu-19.04-disco	192.168.128.119	lrz.small	access_from_windows	Active	nova	None	Running	4 hours, 4 minutes	<button>Create Snapshot</button>
demo-isc	vol:Ubuntu-18.04-LTS-bionic	192.168.128.43	Floating IPs: nvidia-v100.2	lrz_durillo_key	Shelved Offloaded	nova	None	Shut Down	4 months, 3 weeks	<button>Associate Floating IP</button> <button>Attach Interface</button> <button>Detach Interface</button> <button>Edit Instance</button> <button>Attach Volume</button> <button>Detach Volume</button> <button>Update Metadata</button> <button>Edit Security Groups</button> <button>Console</button> <button>View Log</button> <button>Shelve Instance</button> <button>Resize Instance</button> <button>Soft Reboot Instance</button> <button>Hard Reboot Instance</button> <button>Shut Off Instance</button> <button>Delete Instance</button>

# Creating an Instance using the Web Interface

The screenshot shows a web browser window for the lrz cloud provider at [cc.lrz.de](https://cc.lrz.de). The URL bar shows the path `ADS • di57sal`. The left sidebar has a tree structure with categories like Project, API Access, Compute, Volumes, Network, Network Topology, Networks, Routers, Security Groups (which is selected), Load Balancers, Floating IPs, Orchestration, and Identity. The main content area is titled "Security Groups" and displays a table with 8 items. The columns are Name, Security Group ID, Description, and Actions (Manage Rules). The items listed are:

Name	Security Group ID	Description	Actions
InHPC-DE.rv.security_group	d1cdda6d-60b9-4d97-ab6b-1736b3595e80	Security group used for the testing instance containing the remote visualization stack.	Manage Rules
JupyterHUB	8474f2ab-1c1c-4477-aa24-cb4f6f848ef8		Manage Rules
default	2a0b66d5-7a9f-49ae-8d30-f11ddeaf968	Default security group	Manage Rules
example	d926448f-7830-4ec0-8888-a056ff9d4a1a	example	Manage Rules
novnc	65143f52-5536-4e8b-b977-8d0e36f2e056	novnc	Manage Rules
ssh	f4ec8f5f-8797-488a-9cd3-3da12a068cd5		Manage Rules
web	189b2ecd-d0bf-4b9c-a3fc-3f734d57dbe3	web servers	Manage Rules
x11vnc-connections	fb86cef1-828e-4172-a3f5-4346bb75e1e2	Allows direct connections to x11vnc from the client machines	Manage Rules

# Creating an Instance using the Web Interface



# Creating an Instance using the Web Interface

The screenshot shows a web browser window for the LRZ Cloud Control interface at the URL [https://cc.lrz.de/project/security\\_groups/d1cdda6d-60b9-4d97-ab6b-1736b3595e80/](https://cc.lrz.de/project/security_groups/d1cdda6d-60b9-4d97-ab6b-1736b3595e80/). The left sidebar shows a navigation tree with categories like Project, API Access, Compute, Volumes, Network, and Identity. Under Network, 'Security Groups' is selected. The main content area is titled "Manage Security Group Rules: InHPC-DE.rv.security\_group (d1cdda6d-60b9-4d97-ab6b-1736b3595e80)". It displays a table of 7 security group rules:

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	<button>Delete Rule</button>
Egress	IPv4	TCP	4000	0.0.0.0/0	-	<button>Delete Rule</button>
Egress	IPv4	UDP	4001	0.0.0.0/0	-	<button>Delete Rule</button>
Egress	IPv6	Any	Any	::/0	-	<button>Delete Rule</button>
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	<button>Delete Rule</button>
Ingress	IPv4	TCP	4000	0.0.0.0/0	-	<button>Delete Rule</button>
Ingress	IPv4	UDP	4001	0.0.0.0/0	-	<button>Delete Rule</button>

# A Guided Example

**Subtask:** Access the created instance via SSH

- On the Computer Cloud Web Interface
    - **Create a security group that allow ingress connections to port 22!**
    - Add this security group to the instance
  - On your computer
    - Open a terminal application
- ```
ssh -i <path_to_the_pem_file> ubuntu@<floating-ip>
```

After this step, the rest of slides assume everyone is connected via ssh to the created instance

# A Guided Example

- Install pip



```
$ sudo apt update  
$ sudo apt install python3-pip
```

- Install jupyterlab using pip



```
$ pip install jupyterlab
```

- Run jupyter-lab to listen on the private IP of the machine



```
$ jupyter-lab --ip=192.168.3.140
```

You can check the IP using ifconfig (you need to install net-tools)

# A Guided Example

- Connect to she showed URL with a browser. Why is not working?

```
$ jupyter-lab --ip=192.168.3.140
Or copy and paste one of these URLs:

http://192.168.3.140:8888/lab?token=d61e9775638219bcd0260b1cd58467467eed3f7d440ff9
or
http://127.0.0.1:8888/lab?token=d61e9775638219bcd0260b1cd584767467eed3f7d440ff9
```

- On the Compute Cloud Web Interface
  - **Create a security group that allow ingress connections to port 8888!**
  - Add this security group to the instance

# A Guided Example

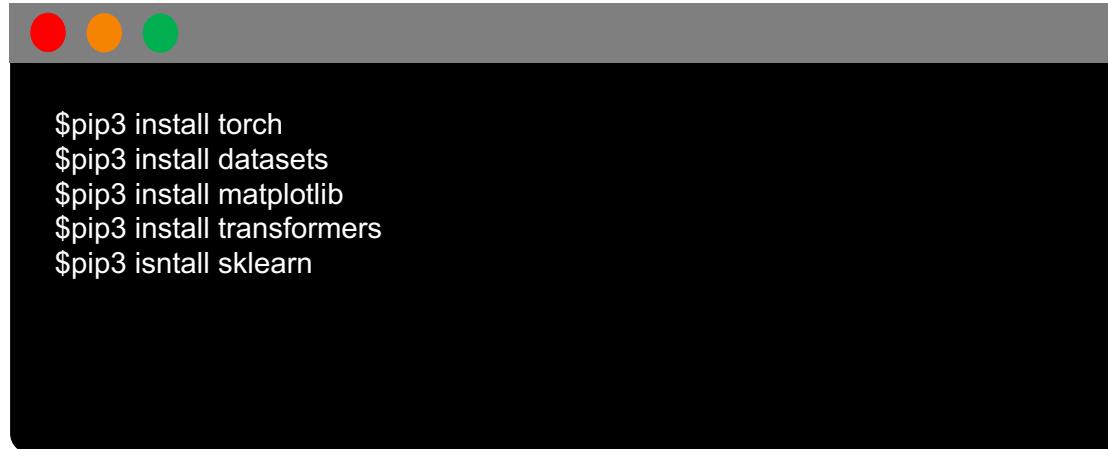
- Connect to she showed URL with a browser. Why is still not working?



- The showed URL is relative to the Compute Cloud instance private network.
  - Substitute in that URL the IP with the floating IP of the instance and try accessing the instance now

# A Guided Example

- Not done: A few libraries are missing for executing our Jupyter Notebook Example



A screenshot of a terminal window with a dark gray header bar containing three colored window control buttons (red, orange, green). The main body of the terminal is black and displays the following text in white:

```
$pip3 install torch  
$pip3 install datasets  
$pip3 install matplotlib  
$pip3 install transformers  
$pip3 isntall sklearn
```

# Summary

- Overview of the LRZ Resources for ML/DL Workloads
- Focus session on the LRZ AI System, a scenario intended for ML/DL training
- Focus session on the LRZ Compute Cloud, a scenario intended for ML/DL development and inference